AD-HOC MULTIPATH ROUTING PROTOKOLLE

Bachelorarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Alican Gecyasar
2006

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

Contents

List of Figures

Acknowledgment

Many thanks to Thomas Staub and Thomas Bernoulli for being very valuable tutors.

Chapter 1

Introduction

A mobile ad hoc network (MANET) is a collection of mobile hosts which interact with each other without the need of any existing infrastructure. Hosts work as routers and interact with each other over wireless links. This particularly means that the network may work as a self-configured stand-alone network. The topology is changing constantly, since the hosts are mobile and moving arbitrarily. There are several ways to set up such a network which we refer to as routing protocols. This work will explicate such different protocols which are being used.

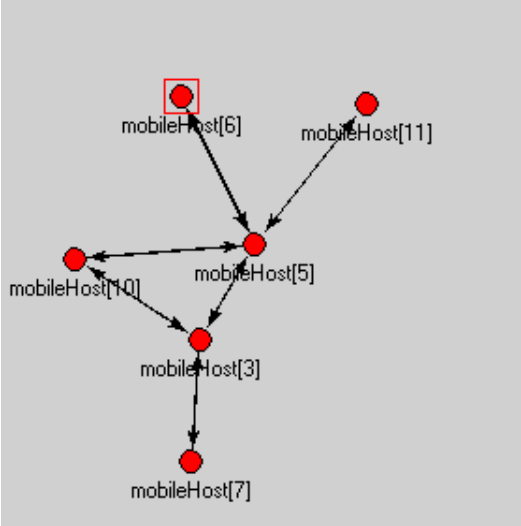

Figure 1.1: MANET

First, we will begin with protocols which support unipath traffic from the source to the destination, whereas the focus will be on Ad-hoc Over Distance Vector (AODV). In the second part multipath routing protocols will be exemplified, whereas we will concentrate on AODV-Multipath and Node Disjoint Multipath Routing (NDMR). OMNeT++ is a discrete event simulation system which was used for the simulations and it is going to be a further topic in this work. Itemized the questions to answer are: Is AODV really worse than AODVM? Which protocol has

the most packet loss? Which of my main three protocols has the best overall performance? Finally I will present my results of the simulations on AODV, AODVM and NDMR.

Chapter 2

Unipath Routing

There are several routing protocols in ad hoc networks. In general they are divided into two groups, the proactive routing protocols and the reactive protocols. Some routing protocols in their belonging groupings are shown in Figure 2.1.

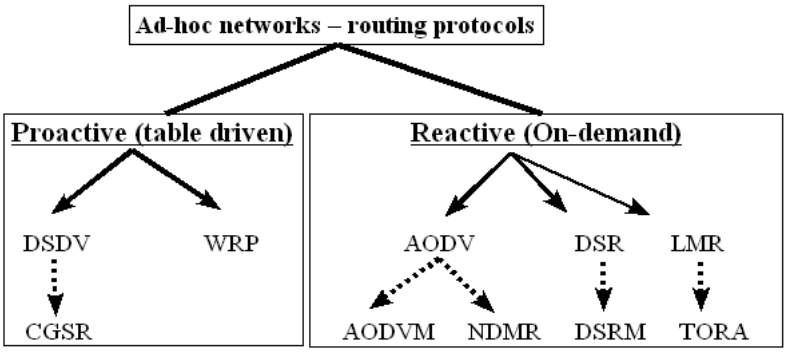

Figure 2.1: Proactive vs. reactive routing protocols

Proactive protocols gather routing information all-time and and the routing information should always be available for each node, so when a node wants to send any messages there should already be a working route to the destination node. On-demand routing protocols are widely used because they consume less bandwidth compared to proactive protocols such as for example DSDV [1] (Highly Dynamic Destination-Sequenced Distance Vector Routing Protocol) or WRP [2] (Wireless Routing Protocol). Therefore we focus on protocols using on-demand routing. The implementation in this Bachelor thesis uses AODV for unipath routing and the two variants AODVM and NDMR for multipath routing. However some more routing protocols are introduced in order to indicate some main differences and to show up the complexity of ad hoc networks.

2.1 DSR - Dynamic Source Routing

As the name states it DSR [3] is a source routing protocol, i.e., the whole route from the source to the destination is included in the packers header. The routing is divided into two parts in DSR. First part is the route discovery to set up a route if there is not already an existing one. The second part is the maintenance of an established connection during a communication between nodes.

2.1.1 Route Discovery

When a source has no entry in its routing cache for a destination, it starts a new route discovery, i.e., it broadcasts a message called RREQ(Route REQuest). The RREQ includes a field recording the traversed nodes. When an intermediate node receives a RREQ it checks if it is already in the route record. If it is, the node ignores the RREQ and does not forward the RREQ any further. To avoid loops in the route, the node also drops duplicate RREQ. If the RREQ is not dropped, the intermediate node checks its own route cache, if there is an existing route to the destination, it sends back an RREP(Route REPly), otherwise it forwards the RREQ to the next node according to the route specified in the message header. If the destination receives the RREQ, it sends back a RREP message. If the destination already has a route to the source in its route cache, the RREP will be sent over this route, otherwise the destination would send the RREP the reverse route back to the source.

2.1.2 Route Maintenance

When a node is trying to forward a message but detects that there is a link break, i.e., the route to the next node is no more established, the forwarding node sends back a RERR(Route ERRor) message towards the source. Whenever a node receives a RERR message, it deletes all his routes containing the broken link.

2.2 AODV - Ad hoc Distance Vector

The AODV protocol is defined by the RFC 3561 [4] written by Charles Perkins and Elizabeth Belding-Royer. Unlike in DSR, AODV uses hop-by-hop routing. That means every node has a routing table which keeps record of routes to other nodes. The table below shows an entry of the routing table in its details.

destId	destS	val	oth	netI	hopCount	nextHop	prec	LifeT

destId	Destination IP Address
destS	Destination Sequence Number
val	Valid Destination Sequence Number flag

oth	Other state and routing flags
netI	Network Interface
hopCount	Hop Count(number of hops needed to reach the destination)
nextHop	Next Hop on the way to the destination
prec	List of Precursors
LifeT	Lifetime (expiration or deletion time of the route)

For a better understanding, especially in the example, I will reduce the table to the following entities:

destId	nextHop	hopCount

2.2.1 Routing

Being similar to DSR, in AODV a route discovery is initiated when a sending source wants a route to a destination but does not already have one in its routing table. To do so a RREQ (as mentioned in DSR) is broadcasted into the network specifying the destination for which the route is requested. When a RREQ packet is received, the node checks, whether it is the destination or not. If it is not the destination, it first checks if it has a route to the destination. If so, the node will send back a RREP along the reverse path. If there is no entry for the requested destination the node keeps on broadcasting the RREQ packet. When the node is the destination it will generate a RREP packet which will also be sent back towards the source along the reverse path. Each node along the reverse path sets up a forward entry in their routing table to the node it received the RREP from. This sets up a forward path from the source to the destination. There is also routing information to gather by the intermediate nodes from the RREP packet on the way back to the source. Intermediate nodes may use those information for further routing table entries to the destination and the nodes between the destination and the gathering node.
Whether the node is the destination or not, when a node receives a copy of the same RREQ packet it will be dropped, without even checking it. This behavior guarantees loop free routes.

2.2.2 Example

In Figure 2.2 the source has no entry in its routing table, so it starts a route discovery by broadcasting a RREQ-packet.
Since nodes A,B and C do not have a route to the destination they keep on broadcasting the RREQ packet. In Figure 2.3 the dotted lines indicate duplicate RREQ-packets that are discarded. Nodes D and E do also not have a route to the destination, so they broadcast again the RREQ-packet as shown in Figure 2.4.

Now the destination have received three times the same RREQ-packet over three different node-disjoint pathes. But as mentioned before, AODV discards duplicate copies of RREQ-packets. That means although we would have found three useable node-disjoint pathes, we discard two of them.

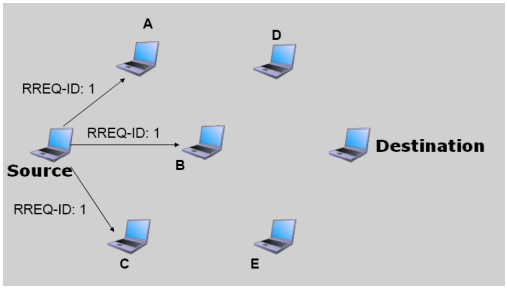

Figure 2.2: AODV - RREQ broadcast by source

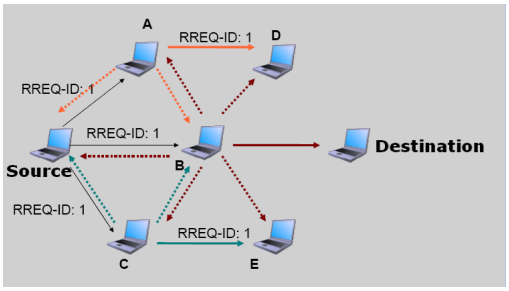

Figure 2.3: AODV - RREQ broadcast by nodes A,B and C

The destination send back a RREP to the first received RREQ. In this scenario that would be node B. The route tables of node B and 'Source' are adapted accordingly as shown in Figure 2.6.

2.2.3 Problems

Obviously, the property of discarding duplicate RREQ-packets does not look appropriate in this scenario. Nevertheless we could imagine scenarios, where it would reduce overhead by reducing the amount of sent RREQ-packets.

2.3 Unipath vs. Multipath

This chapter was dedicated to the two most common ad-hoc network routing protocols with unipath versions. In the next chapter different multipath routing protocols are commented, i.e., protocols which allow more than only one path for routings. The incentive of multipath routings is evidently to reduce the overhead and to guarantee a better network load balancing.

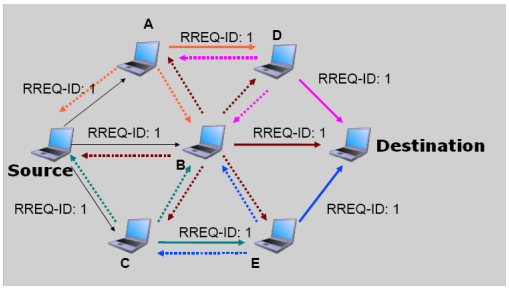

Figure 2.4: AODV - RREQ broadcast by nodes D and E

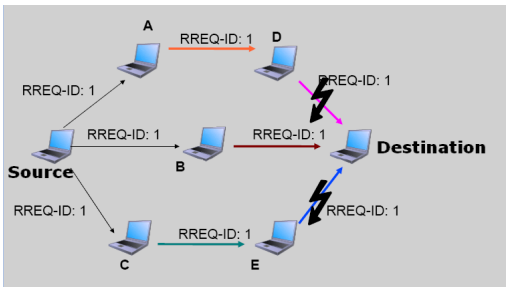

Figure 2.5: AODV - Discard duplicate RREQ-packets

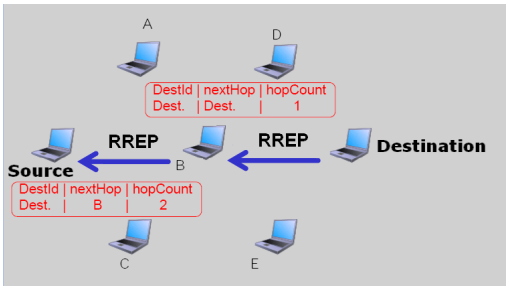

Figure 2.6: AODV - routing table after RREP

Chapter 3

Multipath Routing

Mobile ad hoc networks are vulnerable concerning link breaks, since the nodes are mobile and moving consistently. One should consider using multipath routing, since they are said to be more reliable. In general the following routing protocols should provide multiple paths from the source to the destination. When there are multiple routes between the source and the destination, the sending source might just skip to the second route when the first one is broken without restarting a new route discovery process which is time intensive. At this point we differ between node-disjoint and path-disjoint routes. As shown in Figure 3.1 node-disjoint routes do not have any common nodes in two routes, but in link-disjoint routes two different routes may send over the same node in Figure 3.2.

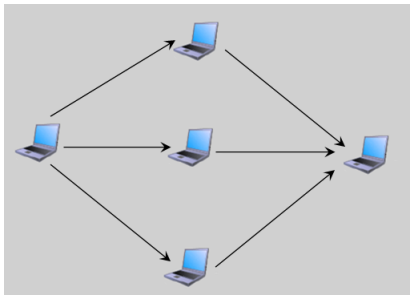

Figure 3.1: 3 node-disjoint routes

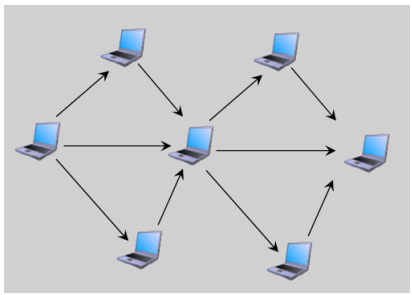

Figure 3.2: 3 link-disjoint routes

MP-DSR and TORA are introduced to point out some difficulties and solutions for multipath routing protocols.

3.1 MP-DSR - Multipath Dynamic Source Routing

MP-DSR [5] is based on DSR, i.e., both protocols work quite similar. MP-DSR uses reliability requirements for end-to-end routes. This reliability is defined by the probability of having a successful transmission between the nodes. Thus some kind of QoS is provided. The calculation of the end-to-end reliability probability is based on the link availability of the intermediate nodes, which is given by their movements. MP-DSR determines two values for

a route: the amount of paths needed to discover and the lowest path reliability that must be fulfilled for a path. When there are less routes between the end nodes, more reliable routes are preferred, accordingly the reliability requirement is higher. After this two values are set, the source node sends out the Route Request (RREQ) for the set amount of paths. Each message contains additional information including the reliability requirement, the path it has traversed, the corresponding path reliability, etc. At an intermediate node, the RREQ-packet is checked whether this message meets the path reliability requirement. If so, the RREQ-packet is dropped, otherwise the intermediate node adds itself to the RREQ and sends out multiple copies of this updated RREQ to its neighbors. The amount of sent RREQ-packets by the source is based on the number of neighbors that can receive this RREQ without failing the path reliability. When the destination receives the RREQ-packets, it selectively chooses multiple node-disjoint routes and sends back RREP-packets accordingly.

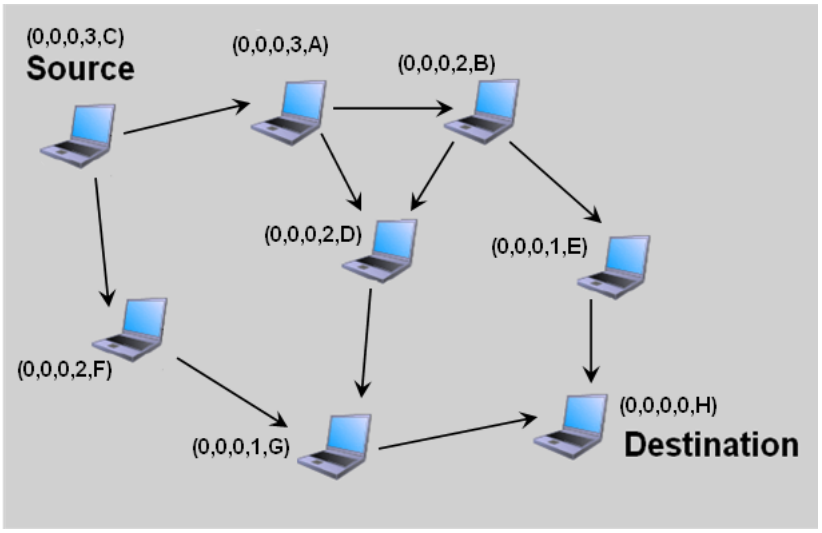

Figure 3.3: Acyclic TORA

3.2 TORA - Temporally-Ordered Routing Algorithm Routing Protocol

TORA [6] is designed to provide multiple routes to the destination. It is based on a link reversal algorithm that discovers and maintains routes through local relaxation of link direction. TORA has three main functions: Establish a route, maintain a route and delete a route. They are executed with the control packets QRY(query), UPD(update) and clear. In TORA each node has a height according to the destination that is set by the routing protocol. The required information to get this height consists of

1. time of a link failure
2. originator id
3. reflection bit indicates 0=original level 1=reflected level
4. integer to order nodes relative to reference level
5. the nodes id

An edges direction between two nodes is givrn by the heights of the two nodes for a given destination. As TORA's edges are directed it becomes an acyclic graph as in Figure 3.3. As node-disjointness has to be given, routes are build accordingly. In this example there would be two routes (A,B,E) and (F,G), where as the route over node F would be the preferred one, since its height is less than the height of the route over node A.

3.3 AODVM - Ad-hoc Over Distance Vector Multipath

AODVM [7] is the first modified version of the AODV protocol which is able to detect multiple node-disjoint paths between a sending source and a receiving destination. AODVM is said to be more reliable and to obtain a better overall performance compared to AODV. Additionally to the routing table (see Figure 3.4), RREQ-tables are introduced (see Figure 3.5) in AODVM.

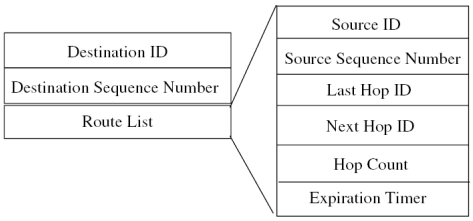

Figure 3.4: Routing table in AODVM

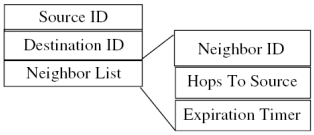

Figure 3.5: RREQ table in AODVM

In the examples the routing table will be reduced to:

Destination	Source	Last hop	Next hop

and the RREQ-table will look like:

Source	Destination	Received from	Hops to Source

3.3.1 Routing

Unlike in AODV, duplicate RREQ-packets are not systematically discarded in AODVM. Instead, whenever an intermediate node receives a copy of a RREQ-packet it records the information in

its RREQ-table as a new entry and rebroadcasts the RREQ-packet. The destination generates a new RREP-packet, containing a new field called ”last-hop-ID”, for every received RREQ-packet and sends it back to the affiliated node. When an intermediate node receives a RREP-packet it first checks if the packet is for itself. If yes the node checks its RREQ-table for an adequate entry otherwise the RREP is dropped. As long as there is an entry in the RREQ-table, RREP-packets are forwarded towards the source. Overhearing RREP-packets for neighbor nodes is one more property which comes with AODVM. I.e. whenever a node 'hears' that one of its neighbors receives a RREP-packet, it removes all the entries in its RREQ-table containing this neighbor, since node-disjoint routes shall be accepted only.
Furthermore, intermediate nodes are precluded from sending a RREP-packet back to the source directly since we want to guarantee node-disjoint routes.

3.3.2 Example

This example is simplified to its basic functions and is more or less self explained. In Figure 3.6 the source broadcasts a RREQ. Node A, B and C record a new entry in their RREQ-table.

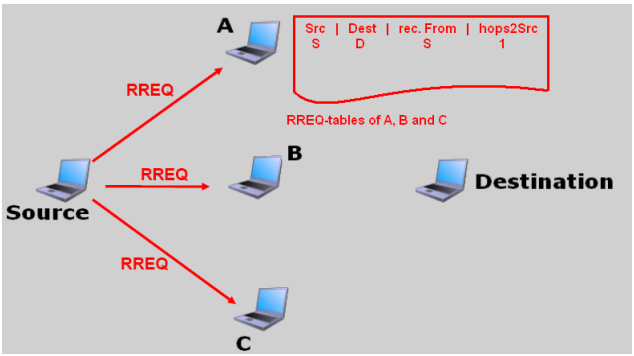

Figure 3.6: AODVM - RREQ by source with the RREQ-tables of node A,B and C

As next step, node B broadcasts the RREQ-packet. Nodes A, C and the destination update their RREQ-tables as in Figure 3.7.

In this scenario everything works step by step for simplicity reasons, this is most likely not the case in real world scenarios. After every intermediate node has broadcasted the RREQ-packet, the destination now reacts on the RREQ-packet sent by node B, thus the destination removes the entry in its RREQ-table and adds the modified entry into its routing-table as pictured in Figure 3.8.

Due to the fact that node A and C hear the RREP-pack sent to node B, they remove their entries according to node B, because we do not want a possibly not node-disjoint route like e.g. (S-B-C-D). The red lines in Figure 3.9 highlight the steps when node B receives the RREP-packet by the destination.

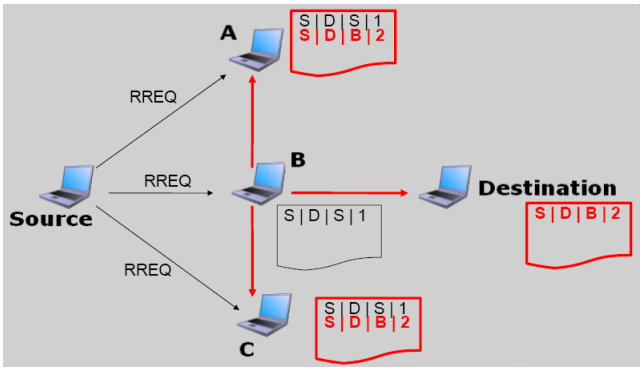

Figure 3.7: AODVM - RREQ broadcast by node B

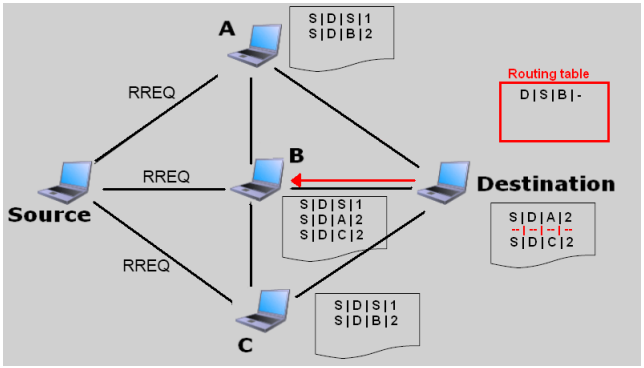

Figure 3.8: AODVM - Destination reacts on the RREQ

The rest of this example is straight forward and the final graph we get is shown in Figure 3.10. Every RREQ-table is empty, and the routing tables are filled.

Now the source node is ready to send data-packets. It is possible to send data packets with a round robin implementation or just over one route and have the other routes as backup routes ready to use. In this thesis they are used as backup routes. This part is going to be explained more detailed in the simulation Chapter 5.

3.3.3 Problems

Like in every multipath protocols there will be less route discovery processes in AODVM. Of course this will lead to less overhead in a network. But in AODVM there will probably be again a higher overhead than in other multipath routing protocols, since AODVM sends a lot of packets around for one route discovery process. Nodes rebroadcast almost every copy of a RREQ-packet they receive.

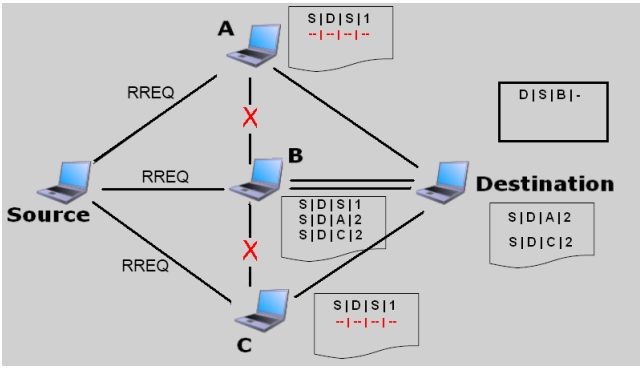

Figure 3.9: AODVM - Overhearing

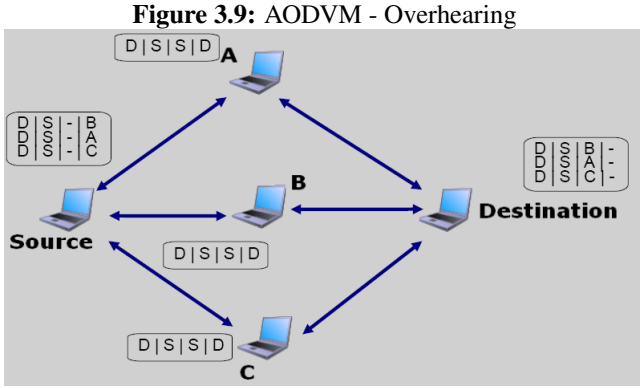

Figure 3.10: AODVM - After the route discovery process

3.4 NDMR - Node Disjoint Multipath Routing

The authors of NDMR [8] named Xuefei Li and Laurie Cuthbert liked the idea of multipath routes in ad hoc networks, but complained about AODVM to have too much memory and routing overhead. Although NDMR is based on AODV, there are a few new properties. Some new features in NDMR include the reverse-route-table, discarding packets when the hop count is too large and the fact that the destination itself decides which route is node-disjoint. A reverse-route-table is basically the same as a route-table but with the other direction. This will be clearer when having a look at the example. What is meant when the hop count is too large is illustrated in Figure 3.11. Only the shortest path between two nodes is accepted.

3.4.1 Routing

In NDMR there is a new field in the RREQ-packet where each node adds itself to. When a source broadcasts a RREQ-packet to detect new routes to a destination it first adds itself to the indicated field in the RREQ-packet. Whenever an intermediate node receives a RREQ-packet it checks if the RREQ-packet has an acceptable hop count considering the 'too-large-hop-count-

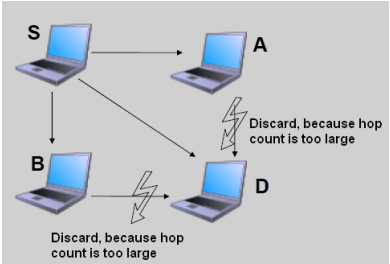

Figure 3.11: NDMR - Hop count is too large

rule'. If the RREQ-packet is not discarded the intermediate node adds itself to the RREQ-packet and rebroadcasts it. After the first RREQ-packet is arrived at the destination, the path which is fully in the additional field of the RREQ-packet is recorded at the reverse route table of the destination and a RREP-packet is generated and replied. When further RREQ-packets arrive at the destination they are first checked if their pathes are node-disjoint regarding the existing routes from the same source. After fulfilling the condition of being a new node-disjoint route and accordingly being added to the reverse route table a new RREP-packet is generated and sent back the route backwards. The RREP-packet has a similar field as in the RREQ-packet which keeps record of the whole path. When transferring a RREP-packet each intermediate node may use the information which is included in the new RREP-packet field. This might reduce the network overload, since there might be less route discovery process. As an example see Figure 3.12, where node B might want to send data-packets to the destination at a later time.

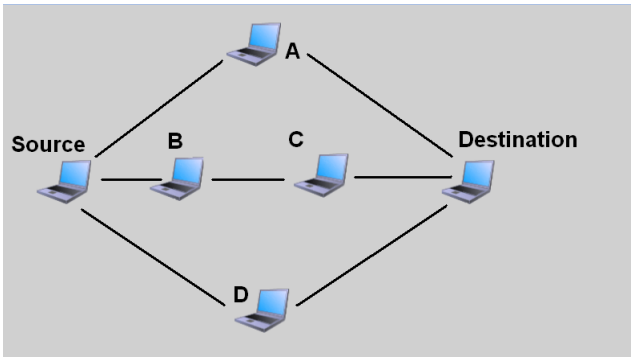

Figure 3.12: NDMR - Profit for intermediate nodes

3.4.2 Example

The same scenario as in AODVM is used to have a direct comparison. The source adds itself to the RREQ-packet and broadcasts it (see Figure 3.13).

Upon receiving the RREQ-packet node B adds itself to the RREQ and rebroadcasts the updated RREQ. Node A, C and the destination receive this RREQ-packet (see Figure 3.14).

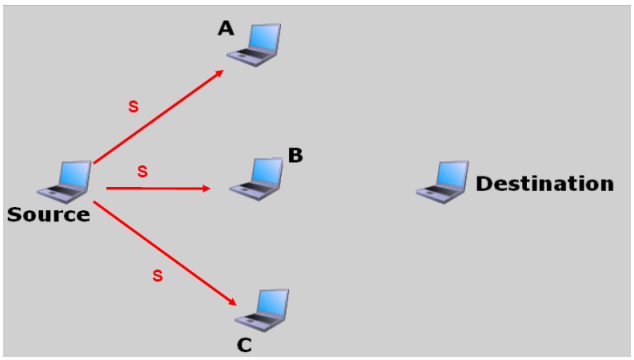

Figure 3.13: NDMR - Broadcast by source

But as shown in Figure 3.15 nodes A and C do not accept the RREQ-packet sent by node B. This means compared to AODVM less overhead, because now, nodes A and C do not send the RREQ received from node B anymore.

The destination adds a new entry in its reverse-route-table as the information is gathered by the received RREQ.

Analogously and with the same reason node B discards the RREQ-packets sent by node A and C. The destination first checks if the received RREQ by nodes A and C are node-disjoint. Because this condition is fulfilled the destination adds this two new route into its reverse-routing-table. In this scenario three RREP are generated and sent back to each node. When the source gets the asked RREP-packet it adds a new entry in its routing table. The graph 3.16 shows the ending of the route discovery process.

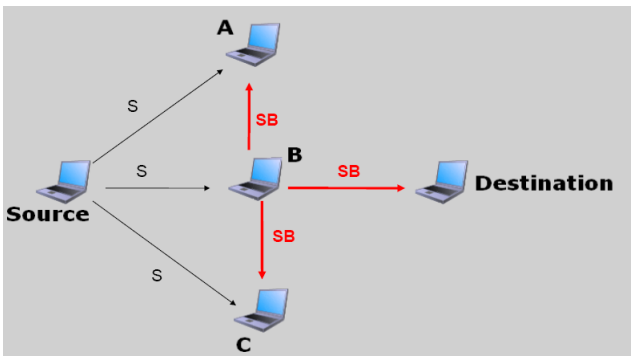

Figure 3.14: NDMR - Updated by node B and rebroadcasted

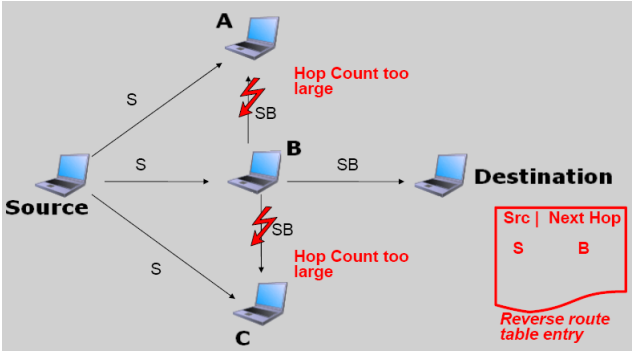

Figure 3.15: NDMR - Discard RREQ's upon too large hop counts

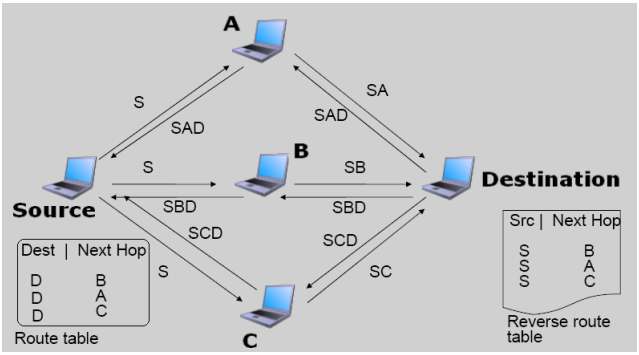

Figure 3.16: NDMR - After the route discovery process

Chapter 4

OMNeT++

OMNeT++ is an object oriented discrete event simulation environment developed by András Varga at the Technical University of Budapest. Its primary use is in simulation of network communications. The developers of OMNeT++ predict that one might use it as well for simulation of complex IT systems, queueing networks or hardware architectures, since OMNeT++ is built generic, flexible and modular. As the architecture is modular, the simulation kernel and models can be embedded easily into an applications.

C++ is the programming language used for the modules in OMNeT++. Compared to NS-2, OMNeT++ is based on only one language, there is no scripting. The two most significant advantages of OMNeT++ are:

1. Documentation - Good extensive documentation and useful supportive user-based mailing-list which is frequented and mostly helpful. In addition there is a wiki where OMNeT++ users share good tutorials. All this can be found at www.omnetpp.org.

2. The OMNeT++ GUI is very helpful especially while debugging and with the inspector windows it is good to understand what is really happening (see 4.1).

One of the most developed supported simulation modules is the INET Framework, which is suited for simulations of wired, wireless and ad-hoc networks. Beyond IP and UDP/TCP there is 802.11, Ethernet, PPP, IPv6, OSPF, RIP, MPLS with LDP and RSVP-TE signalling, NesCT (to run NesC from TinyOS on OMNeT++) and several other protocols. There are also contributed simulation modules which probably cover all the simulation needs.

To create customized simple modules, the following three functions has to be overwritten:

1. initialize() - is often used to read the defined parameters for a scenario from the omnetpp.ini configuration file.

2. handleMessage(cMessage *msg) - is always called when a message is received, this might be any kind of message, e.g., a RREQ-message or as well a self message which prompts the node to move.

3. finish() - is often called upon the end of a successful simulation to log the simulation results.

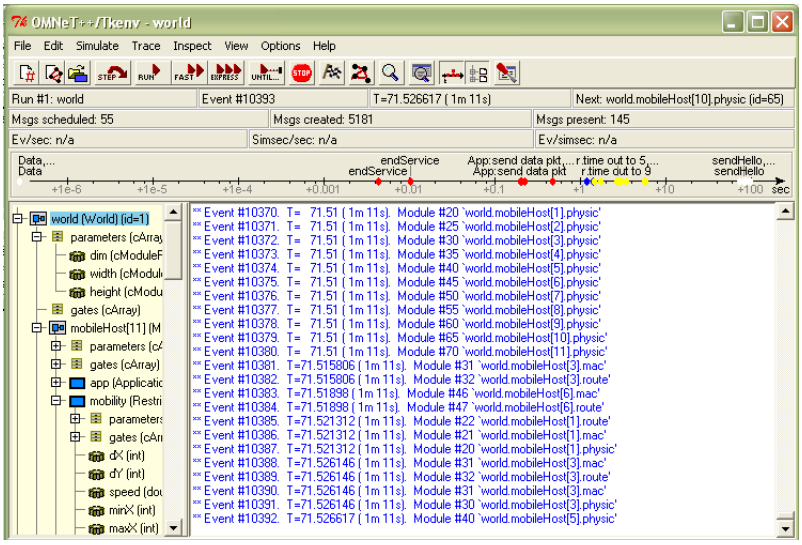

Figure 4.1: OMNeT++ - Main window

The connections between two nodes are called Gates.

The main window (see figure 4.1) contains already almost all of the simulation logs. On the right side every event is recorded. One can also print out values like in C++.

In the simulation window (see figure 4.2) transferred packets may be seen. In this figure a small AODV network is simulated, so the nodes move around and the arrows change regularly.

If one double clicks on one of the moving nodes, the properties of that node appear (see figure 4.3). In this AODV network there are several layers which all can be seen more detailed by double clicking on each of them. In the simulations used for this thesis there are five layers:

1. Application layer - demands a route to send its data to an other node.
2. Route layer - organizes the routes. This is the layer where all the routing protocols are implemented.
3. MAC layer - is used ,e.g., to reduce overhearing in networks.
4. Physical layer - does the actual transmitting.
5. Mobility layer - sends self messages to make the node move around.

For OMNeT++ there is already an implementation of AODV. Its called adHocSim and was implemented by Nicola Concer [9] in the Computer Science Department at the University of Bologna. For this thesis adHocSim was tested if it is RFC-3561 conform and was used as a basis for each implementation.

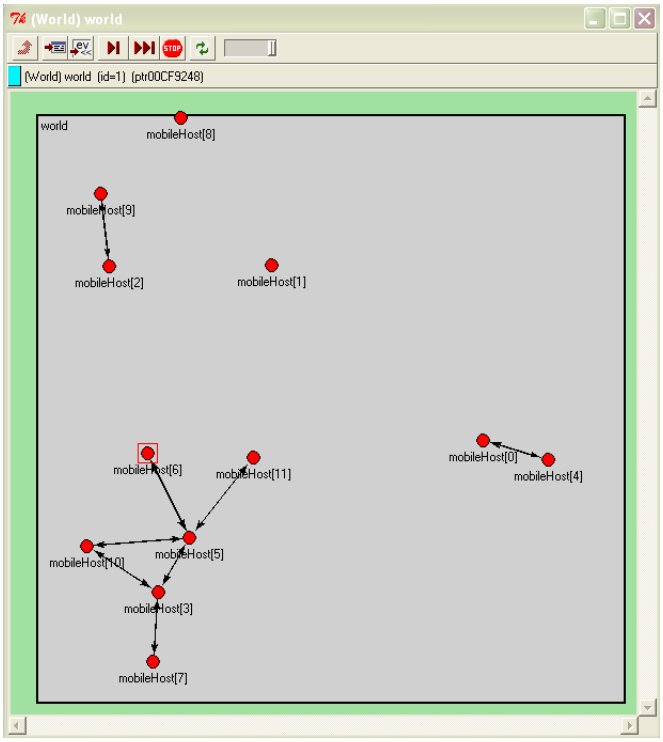

Figure 4.2: OMNeT++ - Simulation window

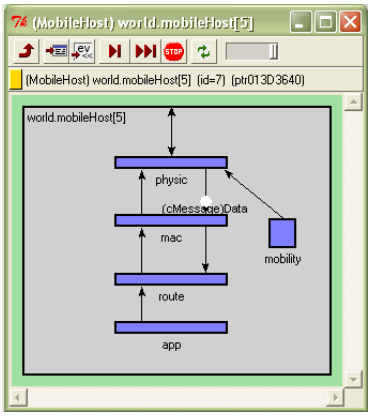

Figure 4.3: OMNeT++ - Node properties

Chapter 5

Simulations

OMNeT++ version 3.2p1 was used to create different simulation scenarios for AODV, AODVM and NDMR.

5.1 Simulation Parameters

The field size is 1000m x 1000m with 50 nodes for the first scenario, for the second there are 75 nodes on an 1000m x 1000m field. Each node uses a 250m transmission radius. There is a scenario, where there is no movement at all, and a second one using random waypoint mobility model [10] with minSpeed 1 and maxSpeed 10. The pause time is set to 0. 512 byte data packets are sent with a ratio of 32 packets every two seconds. The simulations run for 600 simulated seconds. Each data point represents an average of ten simulation runs with identical traffic model, but differently spread initial coordinates. There are two scenarios concerning the amount of sources and destinations. In the first one, there is only one source with one destination. In the second there are three different sources with three different destinations.

5.2 Evaluation

One of our main goals to achieve with multipath routing was to have a better reliability with the backup routes. If every existing routes between the source and destination would have been used, there might have been even a much better performance, i.e., a better throughput for the multipath routing protocols.

Overhead is important as well in networks as it is a factor which states the efficiency of a network. In AODV, there are obviously less sent control packets, since every copy is discarded by arrival.

Especially in fast moving networks time is a very important factor as there might be a lot of route breaks. One should always try to achieve a low latency, since a route may break in a moving network.

5.2.1 No Movement, 50 Nodes

Delivery Ratio - Sent Data Packets vs. Delivered Data Packets

In a simulation where there is no movement, there are surely less packet losses. In AODV and NDMR there are quite good and reasonable results. In NDMR almost every packet is received at the destination. But there is something conspicuous about the AODVM protocol in Figure 5.1. When there are three sending and receiving nodes, the delivery ratio sinks down to 21%. A comparison with the overhead ratio (see 5.2.1) might show the reason for this effect.

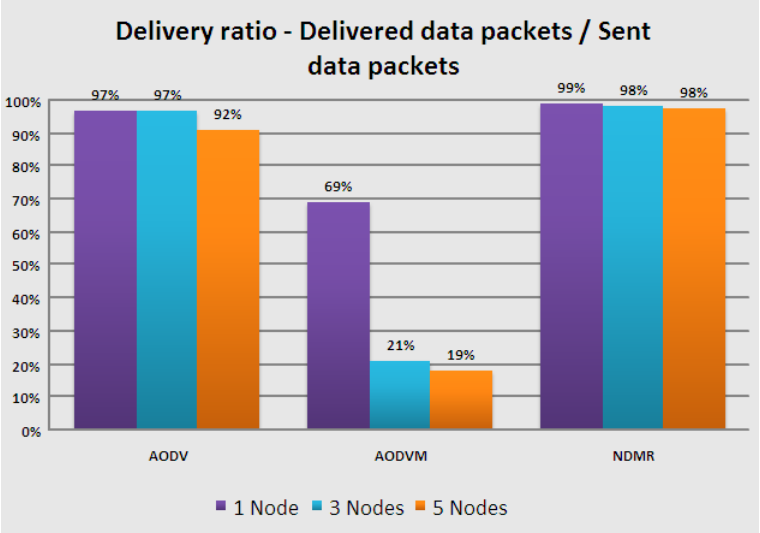

Figure 5.1: Delivery ratio - 50 nodes, no movement

Overhead - Sent Control Packets vs. Delivered Data Packets

There is a very high overhead ratio for AODVM in Figure 5.2. This might explain the low delivery ratio. NDMR has a third of the overhead ratio compared to AODV. The overhead for AODVM appears with the flooding of the network with RREQs. AODVM does not discard duplicate RREQ, instead it even broadcasts almost every arrived RREQ. When there are five nodes who send to five different nodes, the overhead ratio is dramatically reduced. This might be due to the fact that some of the sending nodes may not have to restart a route discovery since they already have routes to the destination achieved by the RREQ of the other sending nodes.

Latency - Delay Time

The time spent for a data transmit in AODVM is most likely not acceptable, where as NDMR and AODV can be compared to each other. NDMR seems to work much faster than AODV, which is probably to ascribe to the lower overhead of NDMR. (see Figure 5.3)

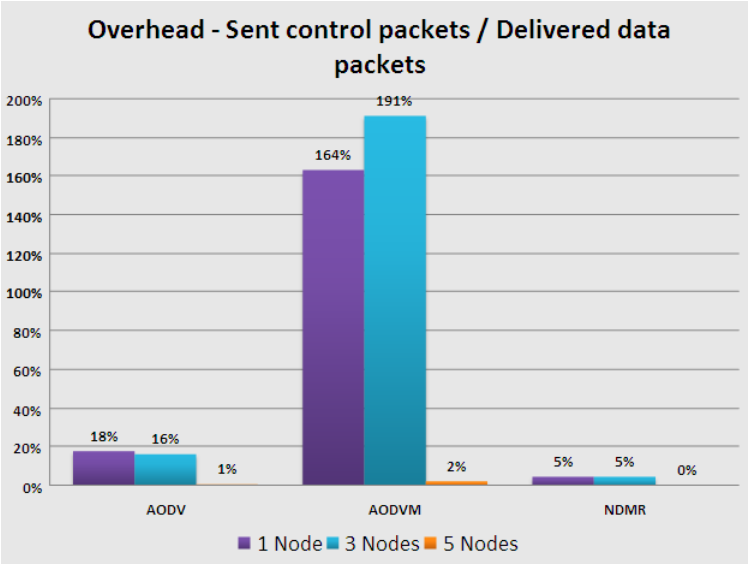

Figure 5.2: Overhead - 50 nodes, no movement

5.2.2 No Movement, 75 Nodes

With 75 nodes in the given scenario field, the average amount of neighbors is approximately 12.

Delivery Ratio - Sent Data Packets vs. Delivered Data Packets

The seen problems with AODVM seem to get worse when there is a higher density in the field. (see Figure 5.4). Where as the both multipath versions seem to result not as good as with only one sender and one receiver, AODV performs better with three active nodes. This might be a result of the capability that intermediate nodes are able to sent back a RREP, if they have an active route to the destination.

Overhead - Sent Control Packets vs. Delivered Data Packets

The overhead ratio of AODVM in Figure 5.5 looks confusing, because it reduces a lot when there are more sender and receivers nodes. But compared to the other two routing protocols, where both have similar effects, it seems like the scenarios must have been the reason for this factor.

Latency - Delay Time

AODV seems to outperform NDMR again in this scenario, when there are multiple sender and receiver. (see Figure 5.6). But if the average is calculated of the amount of nodes, i.e., In NDMR a data transmit takes 0.05sec (= $\frac{0.01+0.09}{2}$) in average and for AODV it takes 0.065 (= $\frac{0.06+0.07}{2}$).

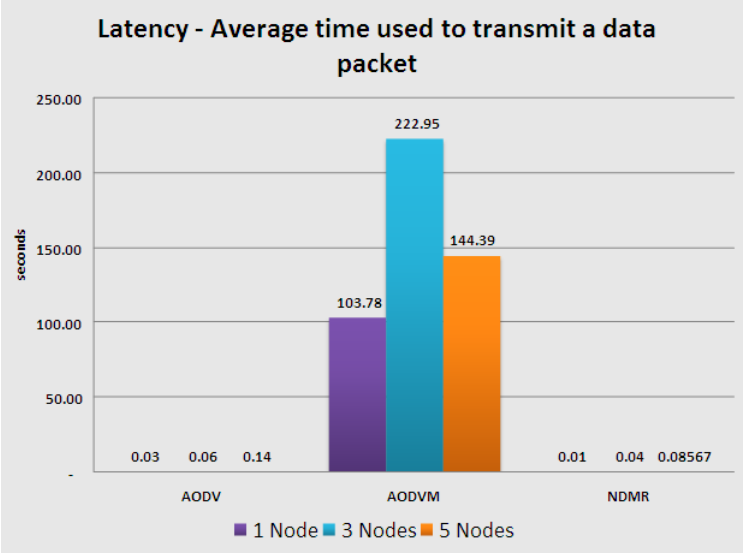

Figure 5.3: Latency - 50 nodes, no movement

5.2.3 Speed between 1 and 10, 50 Nodes

In the following scenarios the nodes are mobile.

Delivery Ratio - Sent Data Packets vs. Delivered Data Packets

In this scenario, NDMR is acting as expected and provides still a very high delivery ratio, where as AODV is clearly no more on top of its performance. With the movement of the nodes the predicted benefits of AODVM seem to disappear once and for all. (see Figure 5.8)

Overhead - Sent Control Packets vs. Delivered Data Packets

The results of the delivery ratio are reflected in the overhead. There are obviously many route breaks, so that AODV has to restart a new route discovery process regularly, which is bonded with more control packets. (see Figure 5.8)

Latency - Delay Time

Figure 5.9 compared to Figure 5.3, where the only difference is the movement, NDMR sends at the appropriate speed compared to AODV. Where as in the non moving scenario AODV outperformed NDMR with three sending and receiving nodes, NDMR catches up in the moving scenario. Supposably this effect appears with the broken links, which happen due to the moving nodes.

5.2.4 Speed between 1 and 10, 75 Nodes

This forth scenario is the last, to show up the better overall performance and reliability of NDMR compared to AODV and AODVM.

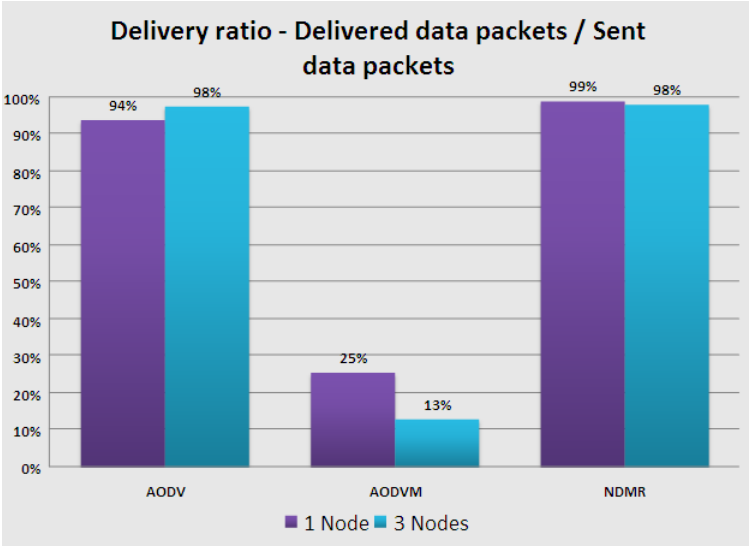

Figure 5.4: Delivery ratio - 75 nodes, no movement

Delivery ratio - Sent data packets vs. Delivered data packets

The multiple routes of NDMR guarantee still a constant around 85 % delivery ratio, which is overwhelming compared to AODV or even AODVM. Meanwhile AODVM seems to be totaly useless in this scenario, as the delivery ratio is less than 5 % in any scenario. (see Figure 5.10)

Overhead - Sent Control Packets vs. Delivered Data Packets

As seen in Figure 5.11, there seems to be a reason why, AODVM gets even worse. The Overhead is amazingly high, that the network is flooded and presumably blocked by control packets. NDMR in the three sender/receiver case has almost a fifth overhead ratio compared to AODV.

Latency - Delay Time

In this final Figure 5.12 NDMR peaks out and outperforms AODV by at least four times within the three sender/receiver scenario.

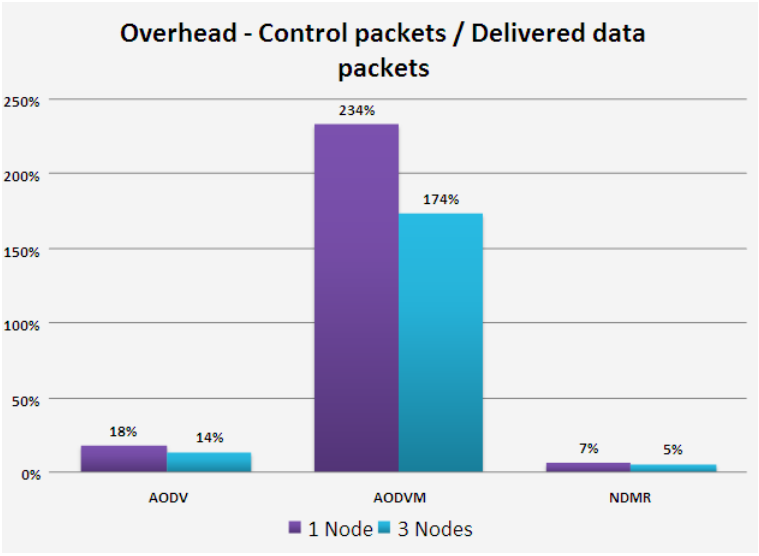

Figure 5.5: Overhead - 75 nodes, no movement

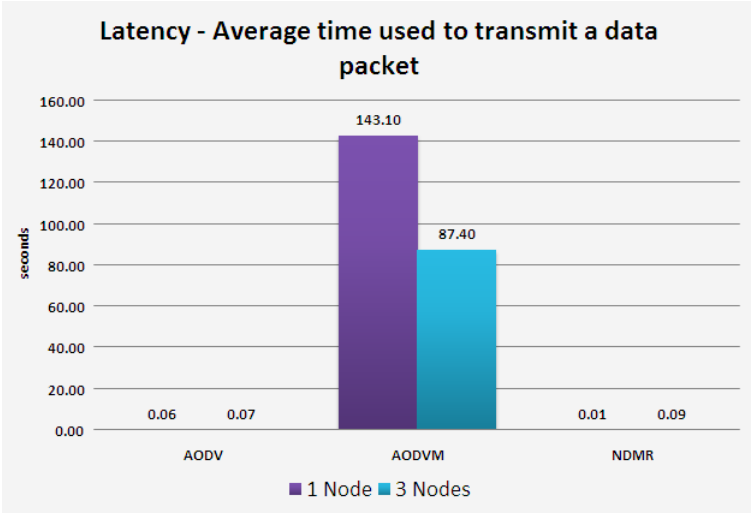

Figure 5.6: Latency - 75 nodes, no movement

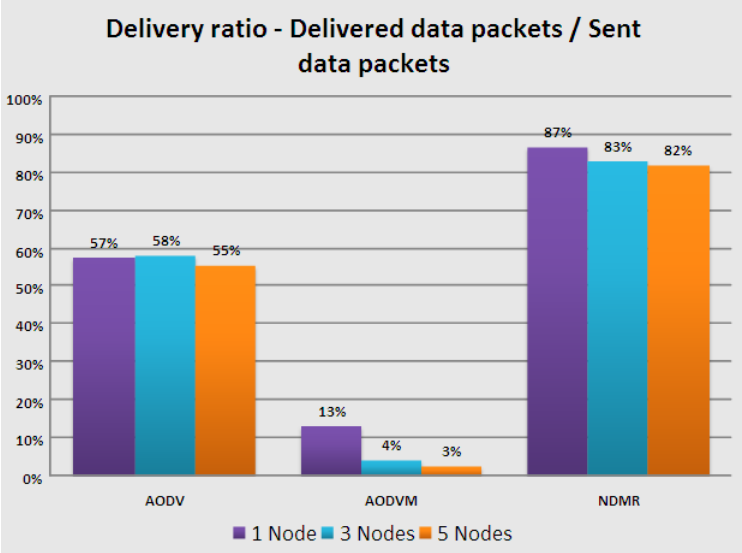

Figure 5.7: Delivery ratio - 50 nodes, with movement

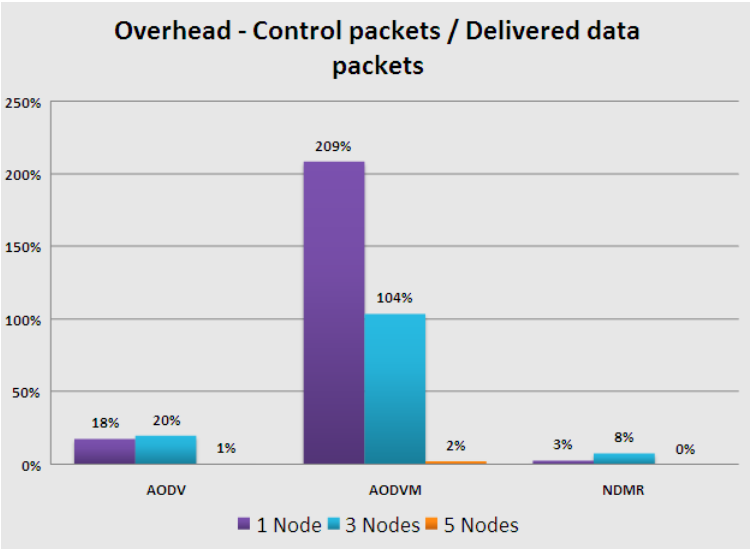

Figure 5.8: Overhead - 50 nodes, with movement

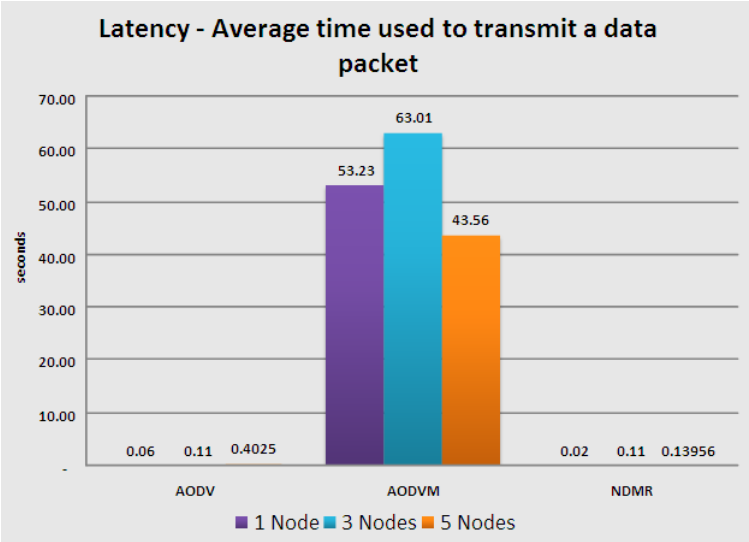

Figure 5.9: Latency - 50 nodes, with movement

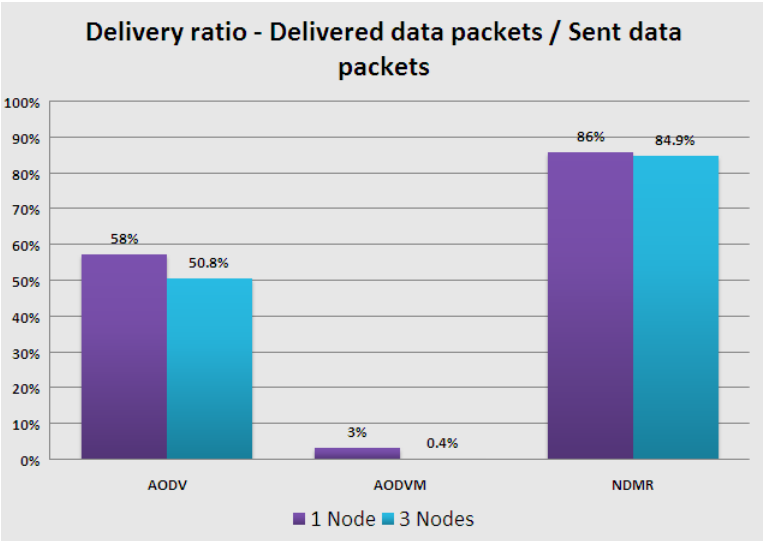

Figure 5.10: Delivery ratio - 75 nodes, with movement

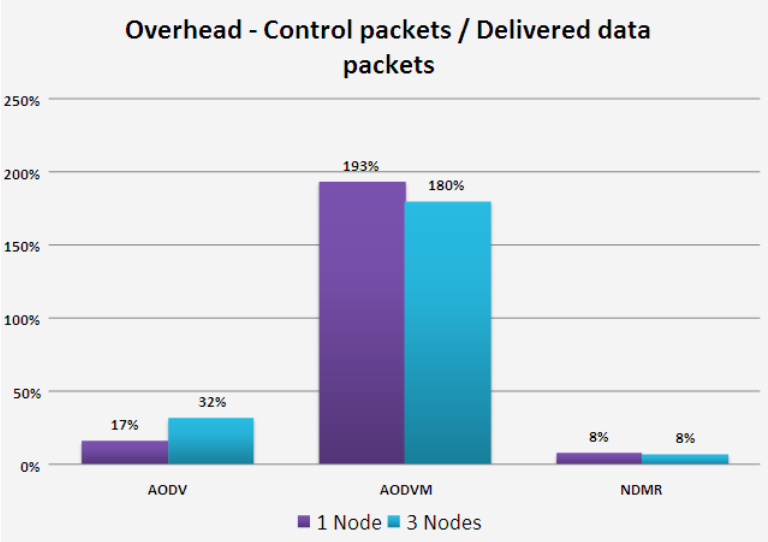

Figure 5.11: Overhead - 75 nodes, with movement

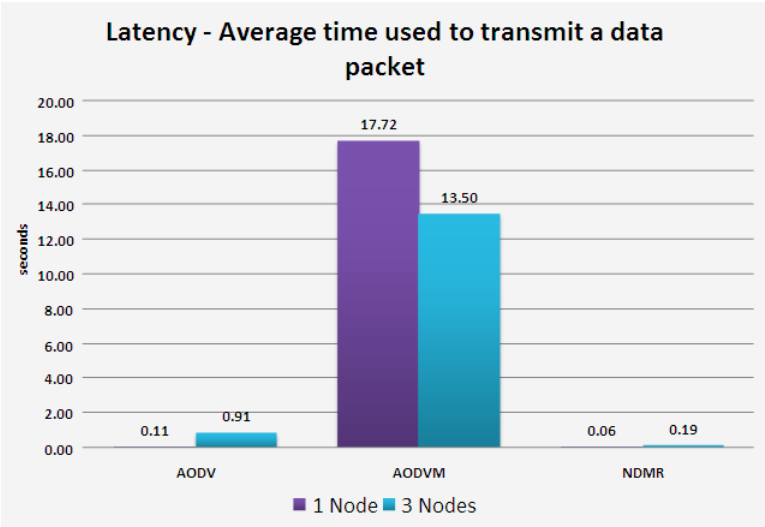

Figure 5.12: Latency - 75 nodes, with movement

Chapter 6

Conclusions

One of the main questions occurs to be, what went wrong with AODVM? We tried to analyze AODVM and believe to have found at least a few points what could have been the reasons for such bad results in our scenarios.

When there are so many neighbors as in our scenarios(around 8 for 50 nodes, and around 12 for 75 nodes), there follows a lot of routes as in Figure 6.1.

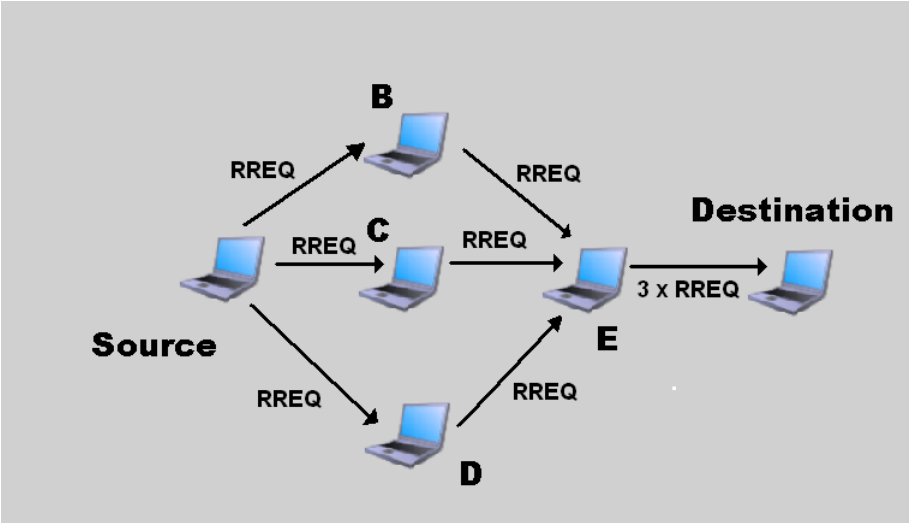

Figure 6.1: AODVM - problem

This is just a reduced figure, for a better understanding. When the node sends out a RREQ, node E receives the same RREQ three times. But as the RREQ does not store the whole path as in NDMR, node E will rebroadcast the same RREQ three times, since it does not know, that the routes may already share a common node.

Imagined a smaller network, AODVM might outperform AODV, since there would be less overhead. Nevertheless NDMR comes off very good in comparison with AODV and AODVM. Thus it is recommended to use NDMR instead of AODV or AODVM.

Bibliography

[1] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing," 1994.

[2] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *Mobile Networks and Applications*, vol. 1, no. 2, pp. 183–197, 1996. [Online]. Available: citeseer.ist.psu.edu/murthy96efficient.html

[3] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353. [Online]. Available: citeseer.ist.psu.edu/johnson96dynamic.html

[4] C. Perkins and E. Belding-Royer, "Ad hoc on-demand distance vector (aodv) routing," 2003. [Online]. Available: http://tools.ietf.org/html/rfc3561

[5] R. Leung, J. Liu, E. Poon, A. Chan, and B. Li, "Mp-dsr: A qos-aware multi-path dynamic source routing protocol for wireless ad-hoc networks," 2001. [Online]. Available: citeseer.ist.psu.edu/leung01mpdsr.html

[6] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *INFOCOM (3)*, 1997, pp. 1405–1413. [Online]. Available: citeseer.ist.psu.edu/park97highly.html

[7] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, "A framework for reliable routing in mobile ad hoc networks." [Online]. Available: citeseer.ist.psu.edu/572538.html

[8] X. Li and L. Cuthbert, "On-demand node-disjoint multipath routing in wireless ad hoc networks," in *IEEE Local Computer Networks (LCN)*, 2004. [Online]. Available: ieeexplore.ieee.org/iel5/9433/29935/01367254.pdf

[9] N. Concer, "adhocsim - an aodv implementation for omnet++." [Online]. Available: http://www.cs.unibo.it/~concer/

[10] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002. [Online]. Available: citeseer.ist.psu.edu/camp02survey.html